an existing transform is explained with respect to figure 9. In another embodiment, the seed icon is icon shifted to form a shifted seed icon $X(B_1\ B_2\ 0)$. The shifted seed icon $X(B_1\ B_2\ 0)$ is exclusive ORed with the icon for the new byte of data $X(B_3)$ to form the second icon $X(B_1\ B_2\ B_3)$. Now the second icon represents an address in the associative memory, so we can determine if there is a match for the data $(B_1\ B_2\ B_3)$. This process then repeats for each new byte of data.

Using this process significantly reduces the processing time required to determine a match. Note that if the process is searching for several three bytes strings it requires the same number of steps as searching for a single three byte string of data. This is because each new data string just represents a different entry in the associative database 26. Whereas standard compare functions would have to perform a comparison for each data string being searched. Thus this invention is particularly helpful where numerous data strings need to be matched.

Often the data strings for which we are searching have differing lengths. In one embodiment this is handled by defining a separate window search size (e.g., $W_{2\text{-}1}$ 24). The two or more window sizes operate completely independently as described above. In another embodiment, the associative database 26 contains a qualified match for a first portion of each the data strings that are longer than the window length. Note in this case the window length (window size) is selected to be equal to the shortest data string being searched. When the process encounters a qualified match, two alternative implementations are possible. In one implementation, there is a pointer 34 associated with the qualified match. The pointer points to a second icon. The process

determines an icon for a next window of data. When the icon for the next window of data matches the second icon a match has been found. Note that this technique can be extended for data strings that have sizes that are many times longer than the window size. However, this implementation is limited to data sizes that are multiples of the window size. This may be limiting in some situations. The second implementation has a match length 36 associated with the qualified match. The match length indicates the total length of the data string to be matched. Then an icon can be determined for the complete data string or for just that portion of the data string that does not have an icon. Using this icon the process can determine if there is match. Using these methods it is possible to handle searches for data strings having varying lengths. This method provides a significant improvement over comparison search techniques, that have to perform multiple comparisons on the same data when differing window lengths are involved.

FIGs. 2 & 3 are a flow chart of the steps used in performing a sliding window search in accordance with one embodiment of the invention. The process starts, step 40, by creating an associative database of a plurality of data strings at step 42. A first window of a data block is received at step 44. The first window of the data block is iconized to form a first icon at step 46. Next it is determined if the first icon has a match in the associative database at step 48. A first byte icon is determined for the a first byte of data in the first window at step 50. An icon shift function is executed to form a first byte icon at step 52. The shifted first byte icon is exclusive ORed with the first icon to form a

seed icon at step 54. A second icon is determine for a second window using the seed icon and transforming a new byte of data onto the seed icon at step 56. At step 58 it is determined if the second icon has a match in the associative database which ends the process at step 60. The process just repeats until the whole block of data has been analyzed for matches. Note the process described above assumes that second window has been shifted one byte from the first window. It will be apparent to those skilled in the art the process can be easily modified to work for shifts of one bit to many bytes. The process described above also assumes that the window is larger than a single byte. However, the process would work for a single byte.

In another embodiment, the process first determines if a single search window size is required. When only a single window search size is required an icon is determined for each of the plurality of data strings. When more than a single window search size is required, a minimum length search window is determined. Next an icon is calculated for each of a first plurality of data strings having a length equal to the minimum length, to form a plurality of first icons. The plurality of first icons are stored in the associative database. Next an icon is calculated for a first portion of each of a plurality of data strings, to form a plurality of second icons. The plurality of second icons are stored in the associative database. An icon is calculated for a second portion of each of the second plurality of data strings to form a plurality of third icons. The plurality of third icons are stored in the associative database. A pointer is stored with each of the second icons that points to the one of the plurality of third icons. Note that in one embodiment a match flag is